Part 2: Advanced Application

• Article: ISO 26262 and E/E software safety risk
ISO 26262 and E/E software safety risk

By Karen Wilhelm, Editor

Programmable and embedded electric/electronic (E/E) systems in automobiles perform safety-critical functions once controlled mechanically. Software in each system that controls its function can contain safety faults that must be discovered and corrected. The complexity of safety-critical software has increased exponentially, making managing safety risk ever more difficult.

One of the things addressed by ISO 26262 is the development of the software in E/E systems and the importance of standardizing development and test methods.

ISO 26262 Part 6, Product development at the software level

The software level of component design is divided into seven phases: initiation, safety requirements specification, architectural design, unit design and implementation, unit testing, integration testing, and safety requirements verification.

In addition to the design of components, the design process itself follows these phases. Among the requirements defined by the design team are modular design, identification of software units, categorizing components, failure analysis, safety mechanisms, and error detection and handling. The design team must select the software development process and tools to be used, and document their choice.

Model-based software design is often selected. While ISO 26262 does not require the use of model-based development, the value and importance of its engineering paradigm is emphasized in Annex B of ISO 26262-6. This means that model-based design and ISO 26262 complement each other in that both approaches aim for high quality development processes for electronic embedded systems.

If models will be used, the team must also implement appropriate software based on them and develop plans for confirming that the implementation behaves as intended. The team also needs to determine the language to be used in the models and in implementation, and select and document any other tools to be used in software development. A number of tools are on the market for design, testing, and validation.

Using the V-Model to guide the software development process

In ISO 26262, a V-Model is often used to represent the development process because testing and verification takes place in reverse order from design and implementation.

The software development phase in ISO 26262 is subdivided into sub-phases as in this V-Model. (In this image, the model begins with “6” which should be considered the first step for the sake of this discussion.) Diagram courtesy of Reactive Systems, Inc.

The model-based development process has several advantages. During the design phase, the model can be tested against the requirements specification, allowing design flaws to be found and fixed early in the development process. Since the models are graphical visual representations of system structure and data flow, they are easier to comprehend than written descriptions. The executable models make it possible to automate implementation testing. When design issues are found, the executable models can be changed and re-tested. Model-based software...
design also facilitates experiments with different design alternatives.

Once decisions about the design process are made, the team can move on to the actual software development. All possible safety impacts of software-based functions, including making a function unsafe or ineffectively handling hardware or software faults must be identified for each software component (step 6 in the diagram). These are the safety requirements to be used throughout the process. An example would be how the component interfaces with other system components in the vehicle and what safety requirements are related to that connection.

After requirements are defined, the high-level architecture for each component is designed (7), and verified for conformance to the safety requirements that relate to it. Likewise, as software unit subsystems are designed (8) and implemented (8), their correctness must be validated. Then in the next step, each low-level software module is unit tested (9). Then systems composed of multiple units are integration tested (10). Testing and verification procedures would include design walkthroughs and inspections, prototyping, and simulations. Finally, verification that its software meets the safety requirements in the actual environment of the component’s intended use is conducted (11).

Example: BMW validates task timing safety
To build ISO 26262 compliant ECU schedules, timing is one performance factor critical for the reliability and safety of increasingly complex systems. The move toward combining more functions into a component makes timing analysis especially difficult. In response, tools for systematically integrating timing analysis into the design process are emerging.

A case from BMW illustrates how one of these tools is aiding the design of safe E/E systems. The BMW 5 series active front steering design required electronically controlled steering intervention for agility and dynamic safety. For this, BMW established a set of scheduling analysis safety requirements.

One of these was stable and predictable timing behavior of the software and the amount of computing power needed for tasks. The scheduling analysis tool used by BMW -- SymTA/S -- was able to identify some rare but critical timing corner cases. The exact times for activation, start, and end of tasks, interrupts, and processes/runnables were recorded. From this, the scheduling analysis tool allowed the reconstruction, charting, and analysis of a real-time situation. The events were displayed as a visual timeline, and a report was produced to show worst-case execution times of those respective tasks, processes/runnables, and interrupts and other timing parameters. Sym-TA/S decomposed the data from the analysis into individual parts (tasks, interrupts, etc.). Then they were re-composed according to all dynamics to identify situations when load and execution times were maximal and check whether enough computing power would be available for them. If not, the engineers made any changes needed to eliminate the shortage of computing power if such an occasion were to occur.

Timing is just one example of the automation of functional safety analysis for the growing number of hardware and software units in today’s vehicles. Safety analysis amid rapid E/E system innovation can only be accomplished if automation tools are available to engineers.

Reusable software to accelerate development

Reusable and modular software designs have long been cited as a way to shorten the development cycle. The Automotive Open System Architecture (AUTOSAR) is a partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry that wants a single open standard for automotive E/E architecture and the development of reusable software components. Such a standard...
platform would enable thorough and consistent safety analysis across the industry and assure safe integration when systems from multiple suppliers are assembled. Reusable software modules would simplify design, coding, testing, and certification for each new vehicle and function, saving a great deal of engineering time and cost. Validated software could be further customized in the development of any new vehicle. It would then be easier to integrate subsystems and show that the entire system is developed according to ISO 26262.

As one step closer, the AUTOSAR Release 4.0 defines many safety features and introduces safety mechanisms and techniques for simplifying ISO 26262 certification and achieving software component modularity and functionality. The AUTOSAR OS safety goal is divided into support for isolated and memory protected regions in which applications can store private data, and a defined, safe scheduling behavior. Memory protection prevents unintended change of private data of Automotive Safety Integrity Level (ASIL) relevant applications by other code. Safe scheduling behavior ensures that tasks activated are actually scheduled by priority. The OS must also be protected from incorrectly functioning applications to cause modifications.

Some of the features of the AUTOSAR framework for safe E/E systems are:

Memory partitioning, which allows fault containment by separating software applications from each other. It can be used for both safety and non-safety applications, especially if combined on the same microcontroller unit.

Time determinism deals with necessary time constraints by synchronizing time-bases for synchronized execution and support. It also detects timing violations and prevents their propagation.

Program flow monitoring controls the temporal and logical behavior of applications by checking the timing and logical order of execution requirements.

Software component end-to-end communication protection provides a library of protection mechanisms, enabling senders to protect data and receivers to detect and handle errors at runtime.

Basic software module defensive behavior prevents data corruption and wrong service calls on microcontrollers lacking hardware support for memory partitioning.

Dual microcontroller provisions enable dual-core implementations that will pave the way for future multi-core implementations. Architectures will be able to detect faults in one core using another independent core.

Overall, by providing ISO 26262 compliant software modules, AUTOSAR will aid the automotive industry by increasing design team productivity and by making vehicles safer.

Trend: Suppliers will make compliance easier

AUTOSAR is not the only way for modular software in E/E systems to be developed and precertified. Suppliers will take care of ISO 26262 compliance in components they sell. Texas Instruments, for example, will assure component-level compliance to ISO 26262 safety requirements from ASIL-A to ASIL-D in development of systems for steering, braking, transmission, electric vehicle battery management and advanced driver-assistance systems (ADAS).

Each component is accompanied by documentation in the form of a design package that includes a Safety Manual detailing product safety architecture and recommended usage, a Safety Analysis Report, and a Safety Report summarizing compliance to standards. The design packages help manage both systematic and random failures. Software included in the package supports documentation and independent third-party evaluation and certification.

www.iso26262-conference.com
Safer driving with safer software

Software-related safety failures have caused recent recalls, and include inaccurate fuel gauges, incorrect airbag occupant restraint control modules, faulty engine control modules, and signal interruption to the steering column. Faults in these E/E systems have endangered vehicle occupants. In addition, recalls are costly and affect an automobile manufacturer’s reputation. Obviously, automakers must take every step necessary to prevent these situations. With functional systems software controlling ever more vehicle operations, strict adherence to the ISO 26262 standard is essential.

About the author: Karen Wilhelm has worked in the manufacturing industry for 25 years. She publishes the blog, Lean Reflections, and writes a regular column for Manufacturing Pulse, a news website for manufacturing executives. Karen is based in Detroit, Michigan, and is a member of the Society of Manufacturing Engineers and the Association for Manufacturing Excellence.

Sources:
